CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Improving navigation in critique graphs
Authors	Genc, Begum;O'Sullivan, Barry
Publication date	2016-11
Original Citation	Genc, B. and O'Sullivan, B. (2016) 'Improving navigation in critique graphs', 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), San Jose, CA, USA, 6-8 November. doi:10.1109/ICTAI.2016.0030
Type of publication	Conference item
Link to publisher's version	10.1109/ICTAI.2016.0030
Rights	© 2016, IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Download date	2026-11-01 20:01:39
Item downloaded from	https://hdl.handle.net/10468/5691



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Improving Navigation in Critique Graphs

Begum Genc
Insight Centre for Data Analytics
Department of Computer Science
University College Cork, Ireland
Email: begum.genc@insight-centre.org

Barry O'Sullivan
Insight Centre for Data Analytics
Department of Computer Science
University College Cork, Ireland
Email: barry.osullivan@insight-centre.org

***Abstract*—Critique graphs were introduced as a device for analysing the behaviour of conversational recommender systems. A conversational recommender allows a user to critique a recommended product with statements such as *"I'd like a similar product to this one, but cheaper"*. A critique graph is a directed multigraph in which the nodes represent products, and a directed edge between a pair of products represents how a user can move from one product to another by tweaking a particular product feature. It has been shown that critique graphs are not symmetric: if a user critiques a product p_i and is presented with product p_j, critiquing product p_j in the opposite manner does not necessarily return product p_i. Furthermore, it might not be possible to reach all products in a catalogue starting from a given product, or as a consequence of a particular critique some products become unreachable. This latter point is quite unsatisfactory since a user would assume that it is possible to explore the full catalogue by critiquing alone. A number of approaches to over-coming this problem have been proposed in the literature. In this paper we propose a novel approach that exploits the critique graph directly. Specifically, the unreachability is a consequence of a critique graph having more than one strongly connected components. We show how the critique graph can be modified in a minor way, thereby modifying the semantics of critiquing for a given catalogue, so that all products are always reachable.**

***Keywords*—Recommender Systems; Critique Graphs; Strongly Connected Components**

I. Introduction

As on-line shopping has become an important part of daily life, customers need the help of recommender systems to find their desired products amongst huge catalogues of possibilities. A recent study that analyses shopping behaviour of on-line customers reveal that they are more likely to buy what the system recommends them [14]. Therefore, designing systems that make good recommendations is essential.

There are two types of users: the first type knows exactly what to buy or which features to look for. However, the second type has only a few criteria in mind and needs to explore and learn about the available products before making a final decision. In this paper, our main target is the second type of users. In order to provide an interactive and educational experience to the user, we make use of conversational recommender systems. Conversational recommender systems engage users in the decision process to produce output that meets with user feedback and requirements [1].

Critique-based conversational recommender systems allow users to interact with the system iteratively to enhance the recommendation process. The user initiates the search by passing some criteria to the system. The system interprets the requirements of the user and finds the best set of products (or a single product). Then, the user can either accept one of the proposed examples or continue the search by critiquing current option(s) until satisfied with the suggestion. There has been extensive research on critique-based conversational recommender systems [3], [9], [12], [13], [11], [16], [10]. A detailed survey of the topic is also available [5]. There are currently three main procedures that have been proposed: user-motivated critiquing, system-proposed critiquing and natural language dialogue based critiquing support. The users can either apply a *unit critique* i.e., requesting an alternative value for a single feature of an item, or *compound critique* by requesting modifications on multiple features at a time.

The concept of critique graphs was introduced in 2008 as a device for analysing the critiquing process, as well as understanding the consequences of a given semantics for critiques in the context of a given catalogue [7]. Critique graphs provide a navigational structure that captures information about closest product pairs in the catalogue, by compiling all products into a large directed multi-graph. In the original paper, the authors introduced the notion of critique graphs and how they can be used to analyse conversational recommenders, as well as offer advice on an ideal set of entry products to present to the user at the outset. However, critique graphs also offer the promise of assisting the designers of conversational recommender systems to design a suitable semantics for critiques that could be well suited to particular catalogues.

One of the main problems in critiquing systems is the *diminishing choices* problem [10]. A user may want to explore the product base first, learn about all possibilities and then choose one of them. Therefore the systems should not eliminate the previously suggested products while making a new suggestion. Another major problem is the *unreachability* problem, where a number of products in the product base can never be reached from a set of products under any critiquing combinations that might be performed [10], [7].

There exists some recovery mechanisms from failures such as allowing previous recommendations if there does not exist any other recommendations [10]. However, this operation requires keeping a list of recommendations and brings along with some cost to evaluate. There are no approaches to resolving unreachability in the literature apart from variations on remembering the history of products presented to the user. Our intention is to pre-process the data so that there remains as little calculations as possible for the recommendation phase.

The novel contribution of this paper is that it uses the graph-theoretic properties of critique graphs to modify the semantics of critiquing to address the navigational maladies in conversational recommender systems. Specifically, unreachability, whereby there does not exist a set of critiques that can move the user from one product to another, is a consequence of a critique graph having more than one strongly connected components. We propose a polynomial-time approach to augmenting a critique graph with a small number of additional edges that can be seen as minor modifications to the semantics of the critiquing process that ensure that all products remain reachable at all times.

The remainder of this paper is organised as follows. In Section II we review the definition and properties of critique graphs. Section III we discuss the reachability problem in more detail and propose our polynomial-time approach to modifying the semantics of critiques for a specific catalogue that ensures all products remain reachable while not compromising on the similarity between products. Section IV presents a set of experimental results that demonstrates that our approach performs extremely well, adding close to optimal numbers of edges, as compared with a well-known graph augmentation method. Finally, in Section V we summarise the contributions of the paper.

II. Preliminaries

We recall the basics of critique-based recommenders and critique graphs [7].

A. Catalogues and Similarity

A catalogue defines a set of products, Π. Each product $p_i \in \Pi$ is defined by a tuple of n attributes $A = \langle a_1, \ldots, a_n \rangle$.

We assume a distance metric, $d(p_i, p_j)$, between products is defined, which enables us to reason about inexact matches between products and user-stated queries for ideal products. We assume that this distance measure satisfies the standard properties of a metric:

1) Non-negativity: $d(x, y) \geq 0$;
2) Identity of indiscernibles: $d(x, y) = 0$ iff $x = y$;
3) Symmetry: $d(x, y) = d(y, x)$;
4) Subadditivity: $d(x, y) + d(y, z) \geq d(x, z)$.

We also assume the distance metric is a weighted sum that is decomposed across the attributes of the product. Specifically, given a pair of products p_i and $p_j \in \Pi$, we define the distance between them as:

$$d(p_i, p_j) = \sum_{a_k \in A} w_k \times d_{a_k}(p_i[a_k], p_j[a_k])$$

where d_{a_k} is a measure of distance on the a_k-th attribute of the catalogue, and the notation $p_i[a_k]$ refers to the value of attribute a_k of product p_i. The quantity w_k is the (non-negative) weight of attribute a_k. We assume distances and weights to be in $[0, 1]$, so the maximum distance between products is in $[0, n]$.

We define the similarity between two products in an analogous manner, as the complement of distance. Specifically, we define the similarity $\sigma(p_i, p_j)$ as:

$$\sigma(p_i, p_j) = \sum_{a_k \in A} w_k \times (1 - d_{a_k}(p_i[a_k], p_j[a_k])).$$

B. Critique-based Navigation

In many applications of recommender systems, the interaction is based on an iterative process during which the system proposes one or more products, and the user provides feedback about his preferences through *critiques*. For example, given a recommendation of a restaurant, a user might critique it by indicating that she would prefer an alternative that is *like this but less expensive*. The two most common forms of critique are *directional* and *replacement* [2]. Directional critiques specify either an increase or decrease in the value of attributes that are either numerical or naturally ordered; a typical directional critique affects an attribute such as price, size, weight, etc. Replacement critiques, on the other hand, specify an alternative value for an attribute taking its values from a finite set; a typical replacement critique would affect a location, a colour, a brand, etc.

Given a product p_i and one of its attributes a_k, we can identify all possible critiques available on that attribute. Directional critiques allow two possibilities: more is better, or less is better. Replacement critiques allow for the substitution of all possible values for attribute a_k other than its current value in p_i, namely $p_i[a_k]$.

The formal semantics of critiquing that we use in this paper is given a product p_i and a critique on attribute a_k, the system responds with the product p_j that is most similar to p_i, but satisfies the critique on attribute a_k; if there are more than one equally most similar products, we consider these also. For example, given a recommendation of a restaurant and a *like this but cheaper* critique, we seek an alternative product that has a lower cost, but is otherwise most similar to the original recommendation. The notion of the set of all possible critiques on an attribute generalizes to all possible critiques over all product attributes, which we will use to define notion of a critique graph.

This approach treats individual critiques independently of any previous critiques a user might have made. However, some approaches consider sequences of critiques as conjunctions of constraints. e.g. [10]. It is possible to also use these richer semantics of critiques here.

C. Critique Graphs

The structure of critique graphs is very suitable for unit critiquing. In a critique graph-based system, one can not specify a certain amount of change in the quantity or ask for a similarity-based search, but instead asks for *more*, *less* or *another value*. Hence, by definition, critique graphs support quality-based unit critiquing. Although it has been shown that dynamic critiquing, which uses compound critiques, has an advantage over unit critiquing in terms of shorter session lengths, system-proposed feature pairs have also been criticised by being too rigid [13], [4].

Given a product catalogue and similarity measures for each attribute, it is possible to pre-compute all possible ways in which the user can critique any product that might be recommended to him. We introduce the notion of a critique multigraph that encodes which specific critiques enable one to move from product to product in the catalogue.

Definition 1 (Critique Multigraph): A critique multigraph, $\mathcal{C}_m(\Pi)$, that is associated with a catalogue of products Π, is a directed multigraph $G = (V, E)$. For each product $p \in \Pi$ we have an associated vertex $v \in V$. Each directed edge $(v_i, v_j) \in E$ is associated with a critique of product p_i whose most similar product satisfying the critique includes product p_j; each edge is labelled with the corresponding critique.

It is sometimes convenient to abstract the critique multigraph to a directed critique graph in which all edges between a pair of vertices are reduced to a single edge with the appropriate direction. A directed edge (v_i, v_j) in a critique graph can be interpreted as stating that there *exists* a critique of product p_i that leads to product p_j.

Definition 2 (Critique Graph): A critique graph, $\mathcal{C}(\Pi)$ of a critique multigraph is a directed graph on the same set of vertices as its corresponding multigraph $\mathcal{C}_m(\Pi)$. All directed edges (v_i, v_j) in $\mathcal{C}_m(\Pi)$ form a single directed edge between (v_i, v_j) in the critique graph $\mathcal{C}(\Pi)$.

An important property of critiquing is that there are situations where some products cannot be reached by any critique sequence from a given product. The notion of *reachability between products* is an important issue that has been studied in [10].

Definition 3 (Reachable Products): Given a catalogue Π, and a pair of products $p_i, p_j \in \Pi$, we say that p_j is reachable from p_i iff there is path from p_i to p_j in the critique (multi)graph $\mathcal{C}(\Pi)$.

The fact that under the standard semantics of critiquing, which we described above, that some products are unreachable from others from the outset, or may become unreachable during navigation, seems a significant problem. The major contribution of this paper is an approach to overcoming this issue by minimally modifying the semantics of critiquing such that all products are reachable.

III. Guaranteeing Reachability in Critique-based Navigation

We seek to guarantee that all products in a catalogue are reachable through critiquing regardless of which attribute the user wishes to critique. We, therefore, define the notion of a decomposed critique graph which considers the critique multigraph on an attribute by attribute basis.

Definition 4 (Decomposed Critique Graph): Given an original critique multigraph $G = (V, E)$, the decomposed critique graph $DG_i = (V, E_i)$ is obtained by decomposing the original critique multigraph into feature based sub-graphs whereby each sub-graph contains edges that only reflect the changes on one specific feature i.

Decomposing into sub-graphs does not affect the size of the critique graph, if the edges of each decomposed graph are abstracted as different layers on the same set of vertices.

Figure 1 illustrates a complete critique multigraph of a catalogue that contains 4 products with 4 directional features on left. This is a randomly created small catalogue to demonstrate the problem. The smaller graph shows the corresponding critique graph, where the set of edges that share the same target and source nodes are grouped together and represented as one single edge. By looking at the critique graph on right, one may argue that the graph is connected and all products are reachable. However, this conclusion is not always correct.

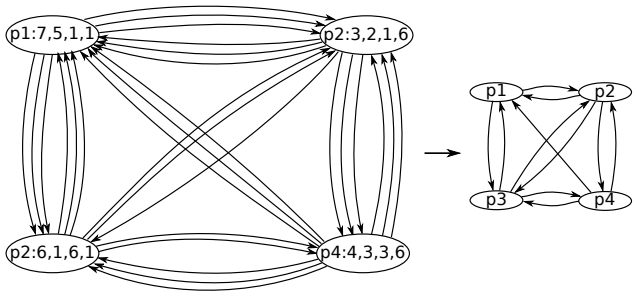


Fig. 1. Sample critique multigraph (left) and its corresponding critique graph (right).

In Figure 2, the sample critique multigraph in Figure 1 has been decomposed. The figure shows one decomposed graph for each feature: in each case the relevant feature is marked as bold letter. The edges between products demonstrate the link between two products.

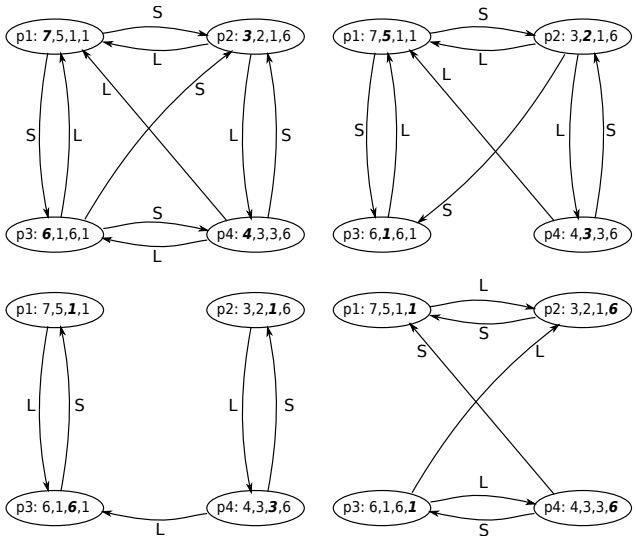


Fig. 2. Feature-based decomposed version of the critique multigraph given in Figure 1. Decomposed graphs belong to: first (top-left), second (top-right), third (bottom-left), and fourth (bottom-right) features, respectively.

In a decomposed graph, an edge $\{p_x, p_y\}$ with the label S indicates that p_y is the most similar product to p_x in the catalogue, when a smaller value of the relevant feature has been requested and L indicates the larger value, respectively. This detailed view reveals more information about how some products are unreachable. For instance, by looking at the decomposed graphs, we clearly see that the customer can reach all products by making critiques on the first and second features. However, there are some unreachable products if she only uses the third or the fourth features.

To be more specific, let's say the system recommended $p2$ to the customer. She requested for more of the third feature and the system proposed her $p4$. However, she wanted even more for that feature and critiqued for more again. The system then recommended her $p3$. However, she thought that the previous

	f1	f2	f3	f4	f5	f6	f7	f8	f9
A	Olympus	7.1	3	22	3	0.8	0.3	Olympus Stylus 730	399.99
B	Fujifilm	5.1	3	10	2.5	0.8	0.3	Fujifilm FinePix Z3	279.99
C	Fujifilm	9	10.7	16	1.8	5.1	1.3	Fujifilm FinePix S9000 9MP	675.99
D	Fujifilm	5.1	3.4	16	3	0.9	0.4	Fujifilm FinePix V10	299.99
E	Kodak	6.1	3	32	2.5	0.9	0.3	Kodak Easyshare V603	299.95

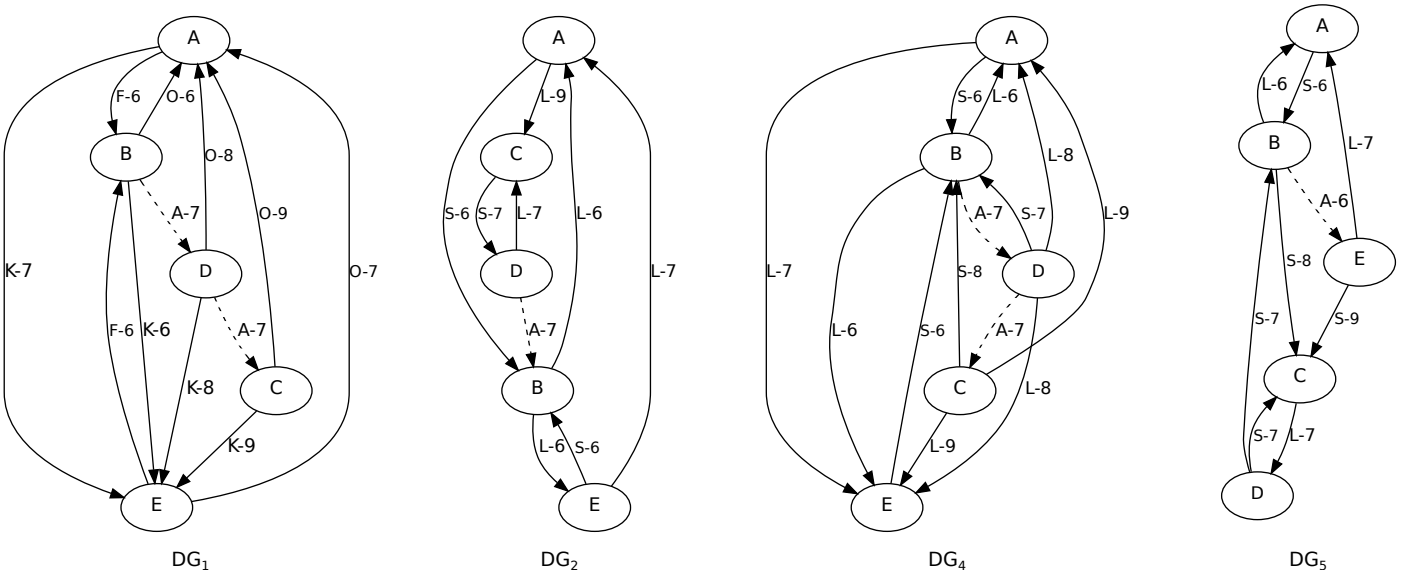


Fig. 3. A 5-product case base and its four decomposed graphs. L and S on the solid edges indicate larger-smaller, and remaining letters represent the initials of the categorical features for readability (i.e., F represents Fujifilm, K for Kodak and O for Olympus) and the numbers indicate the distance. Dashed lines represent some additional edges required for a graph to be fully reachable.

recommendations were better and she critiqued the system for less of feature 3. Unfortunately, the system recommended her $p1$, which is something completely different than what she might have expected. Furthermore, the customer is now stuck in a loop between $p1$ and $p3$ and she can never go back to the previous recommendations $p2$ and $p4$, which she may have liked more. We discuss scenarios in more detail in Section III-A.

A. Demonstrating Unreachability

Conversational recommenders are used as tools to educate users about the products in the catalogue. We assume that the user has some criteria in mind but is not fully aware of her needs. We consider two unreachability use-cases on a simple 5-product camera case base. Figure 3 illustrates 4 out of 9 decomposed graphs of our case base. In order to demonstrate the problem, we will assume that the additional edges are not present in the graph.

1) Structural Unreachability: The user wants a Fujifilm camera and high value for $f4$. By looking at the products, we may infer that C or D are two possible options she is going to be satisfied with.

Assume that the recommendation process starts with product A. The user asks for Fujifilm and B is recommended by the system. The user asks for more of $f4$. By looking at DG_4, we see that only products satisfying this critique are A and E, where both of them are not Fujifilm. If the user critiques A or E by Fujifilm, the system will recommend B again, resulting in a loop. Therefore, the user may think that there are no other products in the catalogue that satisfy her needs. The only way she will be recommended C is by knowing more about other features. In this case, for instance, she is not going to be able to see D, if she is not recommended C first. Hence, no matter what she critiques on features 1 and 4, she may never reach C or D due to the structure of the graph.

2) Diminishing Choices: We now demonstrate the diminishing choices problem, where the user finds the product she likes, but she wants to learn if she can find anything better and continues critiquing. However, at some point she realizes that there are no better alternatives and she tries to trace her critiques back to see a previously recommended product. Due to lack of symmetry in critiquing, there might not exist a path from that product to the one she likes by applying either the reverse critiques or any other combinations.

Assume that the user wants a Fujifilm camera and she is also curious about the range of $f2$ in the catalogue. The system is initialized with product E. She starts making critiques on $f2$. She asks for smaller value, gets recommended B. She likes this one but continues the search. Considering that she can not critique B for a lower value, she knows the lower bound and starts asking for larger values. The system recommends A and E. She critiques A for larger and systems recommends C. At this point, there are no items that have a larger value in $f2$ in the catalogue. Therefore, the user now has the upper bound. She decides that she will track back to a previous recommendation. She asks for a smaller value and D is recommended. From this point on, she is stuck in a loop between C and D in $f2$. She also knows that the other camera she liked was Fujifilm, but both C and D are also Fujifilm cameras. Therefore, she can not critique them for the same value. Hence, she can not go back to the previous

recommendation, unless she memorizes the other features of B or makes random critiques to break the loop.

In these scenarios, the unreachability problem arises because the graphs are not strongly connected. If additional edges are added to the graphs, each individual graph can be made to be strongly connected and all products are going to be always reachable. In the following section, we will explain how to find these additional edges. However, we leave the interpretation of how to present the information using additional edges to the system developer, as this is a user-interface issue. For instance, a developer may design a system such that the system displays the set of products satisfying the critique in the main frame. Then, on a different frame, their neighbours targeted by the additional edges can be displayed as alternative suggestions.

B. Single Strongly Connected Components

The problem of unreachability is a well known problem with critique-based navigation and it has received some attention, e.g. [10] where a memory-based approach is proposed. However, taking a critique graph-based approach to this allows us to take a more formal graph theoretic perspective on the issue. Essentially, the problem of unreachability arises because each decomposition of the critique multi-graph, DG_i, has more than one strongly connected component.

Our approach to guaranteeing reachability is, therefore, to add a small number of additional directed edges to each DG_i such that the *'like this, but satisfying some particular critique'* is relaxed slightly. As we will show below, the modifications to each DG_i are such that we add the most similar edges between products until the graph has one single connected component.

A *strongly connected component* is formed by a subset of the nodes, such that every node in the subset is reachable from all other nodes. In our approach we find the strongly connected components of each decomposed graph by performing a Depth-First Search on each graph in turn using Tarjan's method [15]. After identifying each component, we treat each of them as individual nodes. We update edge connections from node-to-node to component-to-component. Consequentially, we obtain a *Directed Acyclic Graph (DAG)* representation of the strongly connected components. On the resulting DAG, we want to find the unreachable, blocked nodes.

Definition 5 (Blocker Nodes and Components): If there is more than one strongly connected component in a decomposed graph $DG_i = (V', E')$, each node in a strongly connected component c_x becomes a *blocker node* for the nodes in another component c_y if there exists a path from c_y to c_x in DAG $DAG_i = (V, E \subset E')$ associated with feature i. We refer to c_x as a *blocker component* and to c_y as the *blocked component*, respectively.

Figure 4 demonstrates a sample decomposed critique graph of a small instance from a real catalogue. In this example, for instance, product 11 is a blocker node for product 3, which means that if the customer critiques product 3 and obtains product 11, she can never reach product 3 again by making critiques on the same feature. This unreachability can be removed by adding an additional edge from any nodes contained in the strongly connected component SCC (composed of products 1, 2, 5, 6, 7, 8, 9, 10) to product 3, thereby ensuring that this graph contains a single strongly connected component.

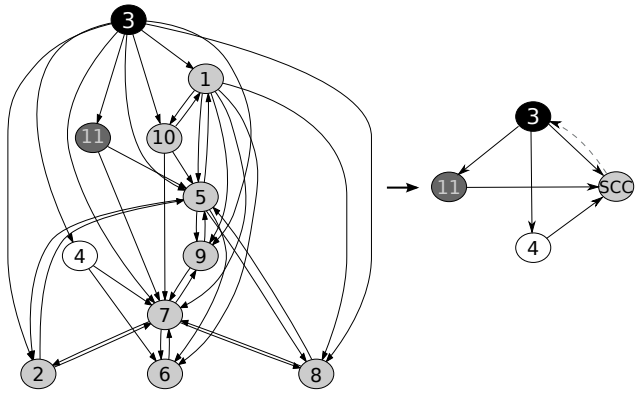


Fig. 4. A sample decomposed critique graph (left), its strongly connected components identified (right), where the required edge to ensure that the component graph contains only one SCC is drawn with dashed lines.

The problem of taking a directed graph with many strongly connected components and adding a set of edges to ensure it has a single strongly connected component is the well-known *strong connectivity augmentation* problem. In cases where the edges are unweighted, finding the minimum number of edges to add can be found in polynomial time [6]. The unweighted case gives us a lower bound on the number of additional edges to be added to each DG_i to guarantee reachability in the critique multigraph. However, for weighted graphs, which we have in this case since the edges have associated measures of similarity between products, finding the set of edges with minimum total weight is known to be NP-hard [6]. Therefore, in the case of catalogue navigation where we want edges to appear between the most similar products subject to satisfying a specific critique, we cannot hope to find a solution that introduces the smallest number of high similarity edges.

A reasonable alternative approach, which can be solved in polynomial time, is to add a minimal, with respect to set inclusion, set of edges such that the resulting DG_i has one single strongly connected component while maximising the minimum similarity of the added edges. More formally, given the directed acyclic graph $DAG_i = (V, E)$ and the set of all possible edges E^*, the objective is to find an $E' \subseteq E^* - E$ such that $CG_i = (V, E \cup E')$ has one strongly connected component and there does not exist an $E^{''} \subseteq E^* - E$ such that $CG'_i = (V, E \cup E^{''})$ is one strongly connected component and the minimum similarity of any edge in $E^{''}$ is greater than that in $E^{'}$. This problem can be solved using an iterative algorithm, very similar to QuickXplain which is generally used for finding maximal relaxations of sets of constraints [8]. The resulting algorithm is presented as Algorithm 1.

The objective of this approach is to add a minimal set of additional edges of highest minimum similarity to achieve a single strongly connected component. First, the algorithm sorts all possible edges E^* in decreasing order of similarity. Mandatory edges are initially identified as the original component graph edges. The algorithm considers each possible augmenting edge of the graph one by one starting from the one that has maximum similarity, until the graph contains one strongly connected component. The last added edge, that

Algorithm 1 Augmentation algorithm

```
1: procedure SCC–CONNECT(V, E)
2:     DAG ← (V, E)
3:     E_man ← E
4:     Set E* to be the set of all possible augmentations
5:     Sort E* by decreasing similarity
6:     while DAG does not have a single SCC do
7:         for e ∈ E* do
8:             add e to DAG
9:             if DAG has a single SCC then
10:                remove e from E*
11:                add e to E_man
12:                break
13:            end if
14:        end for
15:        DAG ← (V, E_man)
16:    end while
17:    return DAG
18: end procedure
```

makes the graph strongly connected is part of the best minimal set and maximises the minimum similarity of edges included. The edge that is added on the first iteration sets the bound on the minimum similarity of the added edges. The algorithm then ensures that only those edges of higher similarity that are necessary to achieve a single strongly connected component are added, thereby ensuring the set of augmenting edges is minimal.

Identifying strongly connected components of a graph has a worst case performance of $O(|V| + |E|)$ using Tarjan's algorithm.

The overall procedure is clearly polynomial in the size of the graph.

IV. Experimental Evaluation

We compare the success of our algorithm with Eswaran and Tarjan's (unweighted) augmentation algorithm [6]. That method computes the minimum number of edges to augment a graph such that there is one strongly connected component. This provides us with a lower bound on the number of edges added in the weighted case. Eswaran and Tarjan's procedure can be summarised as follows:

- Find the number of source nodes (n_i), that have an in-degree of 0 in DAG .
- Find the number of sink nodes (n_o), that have an out-degree of 0 in DAG .
- The minimum number of edges required is equal to $max(n_i, n_o)$.

In our evaluation we have used three well-known catalogues: camera, laptop and travel, which have been previously used in the context of critique graphs [7]. We have removed some product-specific features such as image name and description from the data. Table I provides summary information about the number of products and the number of features used in our experiments. We developed a test tool in Java, using JGraphT library. We perform the tests on a machine with Intel(R) Core(TM) i7-4600U processor with 8 GB RAM.

TABLE I. Details of data sets.

Data set	# of products	# of features
Camera	112	9
Laptop	403	9
Travel	1440	9

There are two alternative standard semantics for constructing the starting critique multigraph: add each product to a *single* example of the most similar products satisfying the critique, or use *all* equally best products. Figure 5 illustrates the effect of two different edge creation policies. Similarly, Figure 6 demonstrates how adding links to all closest neighbours change the diameter of the graph, which provides a measure of how many critiques in the best case are required to find any product from any other.

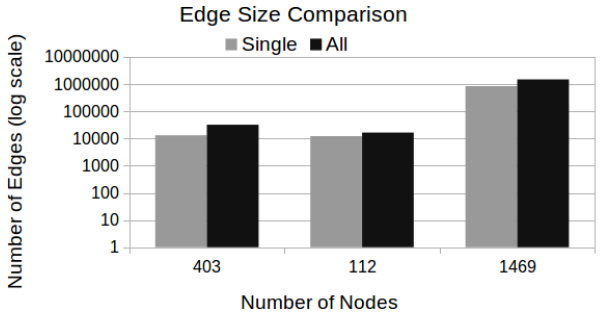


Fig. 5. Number of edges in the critique graph under different edge creation policies.

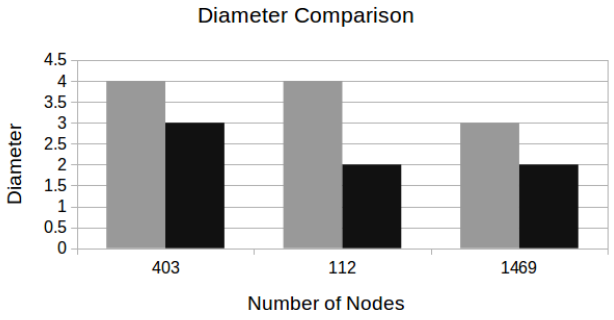


Fig. 6. Diameter of the critique graph under different edge creation policies.

Table II reports our experimental results on the three data sets. For each dataset, there are a set of nine rows, each row corresponding to the decomposition graph DG_i for a feature in the dataset. These appear lexicographically with respect to the order they appear in the data. The *type* refers to whether the critique on that attribute is a replacement (R) or directional (D), depending on whether the attribute is categorical or numeric, respectively.

Number of edges $|E|$ in DG_i denotes the number of edges in the decomposition graph associated with feature i and, therefore, represents the number of possible critiques that can be made on that feature. DAG_i is the connected

TABLE II. CONNECTIVITY ANALYSIS OF GRAPHS AND A COMPARISON OF AUGMENTATION ALGORITHMS.

	type	|E| in DG_i	|V| in DAG_i	$\|E_{man}\|$ in DAG_i	min $\|E"\|$	|E'| added
Camera	R	2370	1	0	0	0
	D	506	8	8	6	6
	D	445	44	64	36	37
	D	577	22	24	18	19
	D	506	24	41	19	19
	D	578	5	4	4	4
	D	519	5	4	4	4
	R	10578	1	0	0	0
	D	446	8	9	5	7
Laptop	R	9139	18	18	16	17
	R	6090	54	63	47	48
	D	1853	28	30	25	26
	D	2617	28	35	20	22
	D	1909	104	157	83	84
	D	2014	43	52	34	38
	D	2214	34	44	27	28
	R	4049	175	184	173	173
	D	2121	29	36	19	20
Travel	R	43430	16	15	15	15
	D	11323	7	7	5	5
	D	10537	73	89	60	63
	R	259053	1	0	0	0
	R	18450	394	492	345	346
	D	9704	206	227	193	195
	R	57623	2	1	1	1
	R	28444	80	95	63	64
	R	1048686	1	0	0	0

component graph of DG_i. Each vertex in DAG_i corresponds to a strongly connected component in DG_i. Therefore, $|V|$ represents the number of strongly connected components in the corresponding DG_i, and $|E_{man}|$ is the number of edges in that component graph. The minimum number of unweighted edges required to compute a single connected component in DG_i is represented by $min|E''|$, as computed using the Eswaran and Tarjan method, which provides a lower bound on the number of edges required to achieve a single connected component and, thereby, guaranteeing reachability to all products during critiquing. Finally, $|E'|added$ represents the number of edges our method, `SCC_Connect` requires to guarantee a single strongly connected component that maximises the similarity of the added edges.

These results show us that we can guarantee reachability by ensuring single strongly connected components by adding a relatively small number of edges to the graphs. `SCC_Connect` performs very well in terms of minimising the number of edges, almost always reaching the lower bound computed using the Eswaran and Tarjan method. The results also show us that decomposed graphs of replacement features are more likely to be strongly connected.

Figure 7 illustrates the success of `SCC_Connect` in terms of augmenting each critique graph while not negatively impacting the similarity between products. Here we plot the worst-case similarity across all edges appearing in the critique multigraph, before and after augmentation. Clearly the augmentation process has had little or no effect on this measurement. Therefore, the additional edges added by the augmentation have little or no negative impact on the quality of the navigation experience of the user.

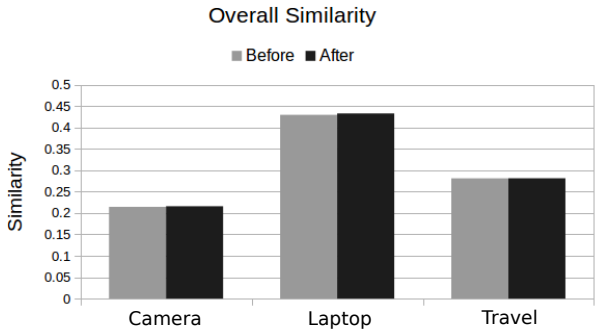


Fig. 7. Comparison of worst-case similarity in the three catalogues before and after the augmentation process using SCC_Connect.

Figure 8 shows the execution time required by `SCC_Connect`. The algorithm scales well with graph size. While this is also a nice property, it is not particularly important since our process is entirely offline and is performed before any interaction takes place between a user and a catalogue.

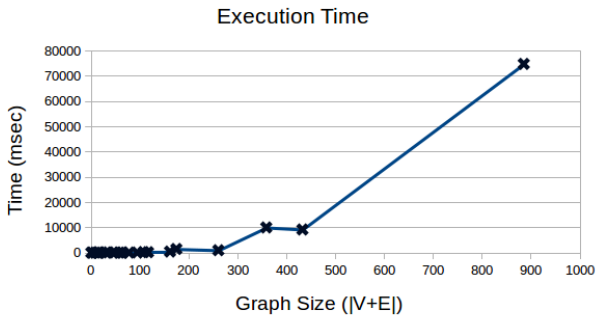


Fig. 8. Execution time of `SCC_Connect`.

V. CONCLUSIONS

The use of critique graphs is a relatively new approach to analysing and designing conversational catalogue navigation and recommender systems. The structure seems to be attractive and promising especially since it relies on an offline analysis. It is suitable for finding closest neighbours of products, as well as compiling all possible critiques that a user might use to explore a catalogue.

Critique graphs can easily explain why during the interaction process that some products become unreachable, regardless of the critiques that the user chooses. This has been a long-standing unsatisfactory property of critiquing since a user would assume that it is possible to explore the full catalogue by critiquing alone. Various ad-hoc approaches to over-coming this problem have been proposed in the literature.

In this paper, building on the graphic-theoretic properties of critique graphs we have proposed a novel approach to resolving the reachability problem exploits the critique graph directly. Specifically, the unreachability is a consequence of a critique graph having more than one strongly connected

components. We have presented a polynomial-time approach to augmenting a critique graph with a small number of additional edges that can be seen as minor modifications to the semantics of the critiquing process that ensure that all products remain reachable at all times.

VI. Acknowledgments

This research has been funded by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289.

References

[1] D. W. Aha, L. A. Breslow, and H. Muñoz-Avila. Conversational case-based reasoning. In *Applied Intelligence*, pages 1–25. Kluwer Academic Publishers, Boston, 2000.

[2] D. G. Bridge and A. Ferguson. An expressive query language for product recommender systems. *Artif. Intell. Rev.*, 18(3-4):269–307, 2002.

[3] R. D. Burke, K. J. Hammond, and B. C. Young. The findme approach to assisted browsing. *IEEE Expert*, 12(4):32–40, 1997.

[4] L. Chen and P. Pu. Evaluating critiquing-based recommender agents. In *AAAI*, pages 157–162. AAAI Press, 2006.

[5] L. Chen and P. Pu. Critiquing-based recommenders: survey and emerging trends. *User Modeling and User-Adapted Interaction*, 22(1):125–150, 2011.

[6] K. P. Eswaran and R. E. Tarjan. Augmentation Problems. *SIAM J. Comput.*, 5:653–665, 1976.

[7] T. Hadzic and B. O'Sullivan. Critique graphs for catalogue navigation. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 115–122, New York, NY, USA, 2008. ACM.

[8] U. Junker. Quickxplain: Preferred explanations and relaxations for over-constrained problems. In *Proceedings of the 19th National Conference on Artificial Intelligence*, pages 167–172, Menlo Park, California, 2004. AAAI Press / The MIT Press.

[9] G. Linden, S. Hanks, and N. Lesh. *User Modeling: Proceedings of the Sixth International Conference UM97 Chia Laguna, Sardinia, Italy June 2–5 1997*, chapter Interactive Assessment of User Preference Models: The Automated Travel Assistant, pages 67–78. Springer Vienna, Vienna, 1997.

[10] D. McSherry and D. W. Aha. The ins and outs of critiquing. In M. M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 962–967, 2007.

[11] P. Pu and B. Faltings. Decision tradeoff using example-critiquing and constraint programming. *Constraints*, 9(4):289–310.

[12] J. Reilly, K. McCarthy, L. McGinty, and B. Smyth. *Advances in Case-Based Reasoning: 7th European Conference, ECCBR 2004, Madrid, Spain, August 30 - September 2, 2004. Proceedings*, chapter Dynamic Critiquing, pages 763–777. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[13] J. Reilly, K. McCarthy, L. McGinty, and B. Smyth. Incremental critiquing. In *SGAI Conf.*, pages 101–114. Springer, 2004.

[14] S. Senecal and J. Nantel. The influence of online product recommendations on consumers' online choices. *Journal of Retailing*, 80(2):159–169, 2004.

[15] R. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1972.

[16] J. Zhang and P. Pu. *Adaptive Hypermedia and Adaptive Web-Based Systems: 4th International Conference, AH 2006, Dublin, Ireland, June 21-23, 2006. Proceedings*, chapter A Comparative Study of Compound Critique Generation in Conversational Recommender Systems, pages 234–243. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.